研究報告

合成データを用いた機械学習に基づく音列分析
Machine Learning-Based Serial Analysis Using Synthetic Data

山本 真幸
Masaki YAMAMOTO
東京藝術大学大学院音楽研究科
Tokyo University of the Arts
Graduate School of Music

田中 翼
Tsubasa TANAKA
東京藝術大学音楽学部音楽環境創造科
Tokyo University of the Arts
Department of Musical Creativity and the Environment

概要

12 音技法をはじめとする音列に基づく音楽作品においては, 原型の音列に対する反行, 逆行, 移高の操作の組み合わせによって得られるさまざまな音列のうち, どれが, 楽曲中のどこに出現するかを特定することがその分析の第一歩となる. しかし, 音列から具体的な音楽へのリアライズの多様性や不完全性から, その特定は困難なことが多く, コンピュータ支援としての手続的アルゴリズムの構築も同様の困難がある. そこで本研究では, そうしたアルゴリズムの明示化の必要のない機械学習に基づく手法を提案する. ただし, 学習データとなる 12 音音楽の分析例は不足しているため, コンピュータで人工的に合成した音列操作ラベル付きの 12 音音楽をデータセットとして代用する. BERT モデルを用いた実験では, 事前学習が音列の検出において有効であることが確認できた一方で, 実際の楽曲データでの精度には課題が残った.

In compositions based on serial techniques like the twelve-tone technique, identifying the occurrence of the transformations of a given original series, such as retrograde, inversion, and transposition, is crucial when analyzing the piece. However, because of the diversity and incompleteness in creating musical content from a series, identifying occurrences of tone rows can be difficult, and constructing procedural algorithms for computer assistance faces similar difficulties. Therefore, this study proposes a machine learning-based approach that does not need explicit descriptions of the algorithmic procedures. However, given the scarcity of analyzed examples of twelve-tone music as the training dataset, we propose to use an alternative dataset comprising artificially synthesized twelve-tone music with labeled operations of series. The experiments employing the BERT model showed that pre-training is an effective means to detect twelve-tone rows. However, when applied to real twelve-tone music, the accuracy was not good, and further improvement is necessary.

1. はじめに

J.S.Bach の多声音楽等にみられるモチーフ音列の逆行形や反行形をとる操作など, 音列操作は作曲における古典的な方法論である. 楽曲分析においては, そうしたモチーフが楽曲中のどこにどのような音列操作を伴いながら登場するかを明らかにすることが楽曲を詳細に理解する上で重要である (Tanaka and Fujii 2014). 20 世紀初頭には, 無調音楽の作曲法として 12 音技法が登場し, 多くの作曲家によってその手法が発展することとなる. それとともに, 音列操作をより数理的に厳密化する思考が発展し, 数理的な音楽理論の発展へとつながった.

12 音技法による初期の作品は, 比較的分析が容易であるが, 時代が進むにつれ, 音列の扱いは複雑化, また作曲家によって多様化していき, その分析は時として困難を極める. したがって, 音列分析を自動化するシステムが構築できれば価値があるだろう. また, ある程度形式化された作曲の方法論である 12 音音楽はそうした自動化の研究の出発点として適している面もあると考えられる. そこで, 本研究においては音列による作曲法の中でも 12 音技法に焦点を当て, その分析の自動化の手法の構築を目的とする.

2. 12 音技法による楽曲の分析

12 音技法においては, 12 半音すべてを 1 つずつ含むように作成された原音列 Prime と, その反行形 Inversion, 反行形 Retrograde, 逆行反行形 Retrograde-Inversion, そしてこれら 4 つについてそれぞれの移高形 12 種類の合計 48 個の音列を用いて作曲を行う. なお本稿では音列の変形を, それぞれの変形の頭文字を取った $O =$

$\{P, I, R, RI\}$ と移高ピッチクラス $t = \{0, 1, ...11\}$ を用いて O_t と表す. これらの楽曲を分析する際, 一般的にはまず, これらの音列がどのように使われているか, 具体的には一つ一つの音符に対して, 音列のどの変形のどの移高形が用いられているか, そしてその何番目かを特定する（以下これをラベリングと呼ぶ）.

ここで実例を挙げる. A. Schönberg の《弦楽四重奏曲 Op.37》の冒頭部は図 1 のように音列分析される. ここでは, P_0 音列が第 1 ヴァイオリンにおいて旋律として用いられ, 第 2 ヴァイオリン・ヴィオラ・チェロはおおよそ小節ごと（譜例における実線の囲みごと）に旋律の不足したピッチクラスを補う形で音列の連続した 3 音による和音を形成している.

音列使用にはさまざまな方法論が存在し, 一般的に用いられているものから, 個々の作曲家独自のものまで幅広い (Leibowitz 1949)(Boulez 1963). 以下にいくつかその例を挙げる.

- **音列の水平的使用：**
  図 1 における第 1 ヴァイオリンのような用法. 原始的な手法で, 音列を横に並べて使用する.
- **音列の垂直的使用：**
  図 1 における第 2 ヴァイオリン, ヴィオラ, チェロのような用法. これも原始的な手法で, 音列を縦に並べて使用する.
- **音列の一部欠損：**
  一度用いられ始めた音列は, その 12 音全てを使い切るのが通常であるが, 一部が欠損することがある.
- **音列の部分入れ替え：**
  音列内のピッチクラスはその出現順序を守るのが通常であるが, 一部入れ替わることがある.
- **音列のオーバーラップ：**
  本来であれば 12 音全てを使い切ってから, 新たな音列を導入するが, 終了を待たずに新たな音列がオーバーラップしてくることがある.

音列からのリアライズの方法論は作曲家によって異なり, ここに挙げた用法はその一部でしかない. こうしたリアライズの多様性, 不確定性が分析の困難さを生む.

3. 提案手法

3.1. コンピュータを用いた音列分析の手法

従来こうした 12 音音楽の音列分析は手作業で行われてきたが, ここではその分析をコンピュータを用いて行う手法を考える.

まずは, 12 音音楽中のある分析対象に存在する音列を探索するルールを記述する手続的プログラミングを行い利用する方法があげられる. しかし, セクション 2 で述べたように音列からのリアライズは作曲家によって無数と言って良い方法論が存在し, それら全てを網羅した上でプログラムの記述を行うのはほぼ不可能であると言って良いだろう. さらにラベリング以前に分析対象を決定するセグメンテーションの問題を先に（あるいは同時に）解かなければいけないが, それが難しい. また, もともとこうしたラベリングは作曲家による生成（リアリゼーション）に由来し, そのプロセスによって自動的に決定されるものであり, 分析によって正解が定義されるものではない. したがって, 分析はリアライズされたものから元のラベルを推定する逆問題であるといえ, こうした逆問題においては推定の不確実性という困難が付きまとう. より具体的に言えば, 音列の一部が欠損していたり, 垂直的和音として現れ, 音列内の音の順番が特定できなくなったりする場合, 音列の候補が複数生じるかもしれず, そのどれを正解と定義するかは不確定となる.

そこで, 手続的なプログラムによる分析ではなく, 機械学習によるラベリングを行うことが考える. 具体的には, ある 12 音音楽を固定長のスコープで切り取ったシーケンスを入力し, そこに音列が含まれているかどうかを判定することを考える. つまり, 切り取ったシーケンスを入力とし, 12 音音列 P_0 と原音列, 逆行, 反行, 逆行反行とその移高形計 48 種類それぞれの操作 $O = (o_1, o_1, \ldots, o_{48})$ に関するラベリング結果 $Y = (y_1, y_1, \ldots, y_{48})$ を出力とするようなニューラルネットワークを構築する手法が考えられる（各音列操作 o_k がスコープ内に存在するとき y_k は 1, そうでないとき 0）. この場合, 音列のリアライズの多様性に対して明示的なプログラムを書かなくて良いという利点がある. ただし, そこで問題となるのはデータセットの入手である. 12 音技法による楽曲は, 調性音楽と比較して研究対象となることが少なく, パブリックドメインとなっていない楽曲も多いため, MIDI などコンピュータで扱いやすい形式にされている例が少なく, 正解ラベルが入手できる楽曲も少ない.

3.2. 人工的な楽曲生成によるデータセットの構築

そこで, 本研究が提案するのは, プログラムによる 12 音音楽の分析ラベル付きのデータセットの人工的な生成である. これは, ランダムに生成した 12 音音列に対し, ランダムに音列操作を施し, 12 音音楽をリアライズする生成プログラムによって擬似的な楽曲データを大量に生成することで, 音列ラベルと楽曲のペアとなったデータの不足を解消するアプローチである. 分析によるラベリングにおいては, 作曲者にしか本当の正解がわからないが, 生成するのであれば正解が自動的に決まるため, 判定の客観性が確保できるのも利点の一

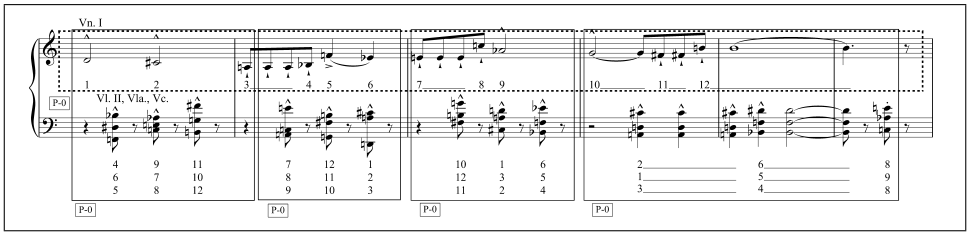

図 1: Schönberg 《弦楽四重奏曲第 4 番 op. 37》第 1 楽章冒頭部（略譜）

つであろう. この意味では, 12 音音楽においては分析よりも自動生成の方が容易である.

4. 人工的な楽曲データの生成方法

楽曲データの生成には python 用ライブラリ music21 を用いる. 前述の通り, 12 音技法による楽曲は, 分析する際の困難はあるが, 擬似的に生成するのは容易で, 大量の合成データを得ることが可能である. 今回は以下のステップで生成を行なった.

(i) ランダムに原音列 P_0 を生成し, それに基づいて, 48 種類の音列を用意する.

(ii) 48 種類の音列からランダムに選ばれた音列を基に音符を配置し, それらすべての音符に使用した音列のラベリングを行う. 学習の際には発音のみを考慮するため, 音価は全て, クオンタイズを行う単位音符としての 16 分音符で統一する.

ただし, (ii) のステップにおいては, 以下をランダムに実行する.

- 音列の水平（旋律）/垂直（和音）使用の切り替え
- 休符の挿入
- 音列の一部欠如
- 音列の部分入れ替え

さらに, 楽曲全体の声部数も 1〜4 で指定可能にした. これにより別の種類の音列が並行するという困難なタスクへの対応も目指す.

5. 機械学習の手法

5.1. BERT の導入

BERT *Bidirectional Encoder Representations from Transformers* (Devlin et al. 2018) は, Transformer を用いた自然言語処理分野における機械学習の手法であり, そのステップは大規模なコーパスを用いた教師なし学習を行う事前学習と, それぞれのタスクに合わせた事後学習 (ファインチューニング fine-tuning) に分かれる.

5.2. 楽譜データのトークン化

自然言語処理分野における BERT への入力にはトークン化された文章が用いられる. 音楽分野における楽譜のトークン化については, MuseBERT(ZiyuWang and Xia 2021) や MIDIBERT(Chou et al. 2021) などの先行研究があるが, ここではより 12 音音楽に特化したトークン化の手法を考える.

まず, 楽譜データについて以下のような処理を行う.

1. 単位音符 (今回は 16 分音符) でクオンタイズする.
2. 全ての音価を単位音符にする. したがってここでは発音のみが考慮される.
3. 全ての声部を重ねる.
4. 発音のない時刻は削除し切り詰める.
5. 20 シーケンス（単位音符 20 個分) ごとに切り出す.

こうして得られたそれぞれの楽譜データについて, "[NOTES]"トークンと, MIDI ノートナンバー 0〜127 に対応する"0"〜"127"トークンを用いてトークン化を行う（図 2）. これにより, 単音, 和音どちらの場合でも同様に扱うことができる. また, BERT に入力する際は, 以下の特殊トークンも用いられる.

- [CLS]: 入力トークン列の先頭につけて, 事前学習における NSP タスクや, 事後学習における分類タスク等で用いる.
- [MASK]: マスクトークン. 事前学習における MLM タスクにおいて用いる.
- [SEP]: セパレータトークン. 文と文（音列情報と楽譜情報）の区切りを示す.
- [PAD]: パディングトークン. 入力が常に最大トークン長になるようにパディングする際に用いる.

5.3. 事前学習

自然言語処理における BERT の事前学習には, MLM (Masked Language Model) と NSP (Next Sentence Prediction) が用いられる. MLM では, 入力トークンの一部

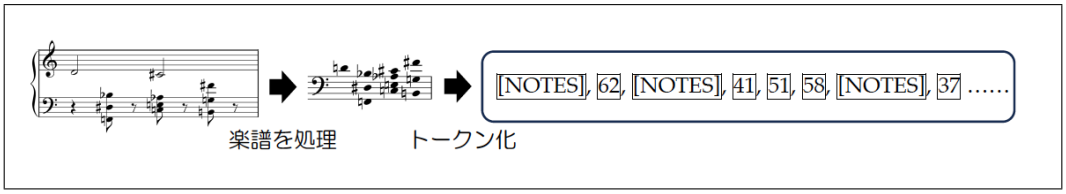

図 2: 楽譜のトークン化

がマスクトークンに置き換え, 元のトークンに復元するというタスクを学習する. 他方 NSP では, 2 つの連続した文章のペアを用意し, そのペアの一部のうちの片方を別の無関係な文章に置き換えて, それらの文章が連続した文章であるか, というタスクを学習させる. これにより, MLM においては文章の厳密な意味を, NSP においては文脈を BERT に教師なしで, 学習させることが期待される. ここでは, 12 音技法の音楽構造, 文法理解を行うためのタスクを考える.

5.3.1. MLM

本研究においても, MLM タスクはマスクトークンへの置き換えを行う. ここでは Devlin らの手法と同様に, 切り出された楽譜から作られたトークンの 15% をランダムに選び, その中の 80% をマスクトークンに, 10% をランダムに選ばれた別のトークンに, 残りの 10% をそのままにして, BERT モデルに元のトークン予測を行うタスクを損失関数として Cross Entropy Loss を用い学習させる.

5.3.2. NSP

NSP タスクでは, 連続した文章ペアではなく, 元となった音列情報（P_0, I_0, R_0, RI_0）と切り出された楽譜情報をペアにし, これらのペアのうちの 50% をランダムに選び, 無関係な音列情報に置き換えて, BERT モデルにそれらのペアが関係あるか, 無関係かを分類するタスクを損失関数として Cross Entropy Loss を用い学習させる.

事前学習においては MLM タスクと NSP タスクを同時に学習させる (図 3). 入力に用いた合成データの生成条件, ならびに BERT モデルのパラメータ数は以下の通りである（表 1, 2).

5.4. 事後学習

前述の手法で事前学習を行った BERT モデルに, ある楽譜シーケンスにおける音列操作を予測するタスクを事後学習させる. 入力には事前学習と同様に, トークン化された音列情報と楽譜情報を与え, BERT の最終層における [CLS] トークンの埋め込み表現を 2 層の全結合層に通して出力 $Y = (y_1, y_2, ..., y_{48})$ を得る. 正解データには楽譜生成の際に行ったラベリングから得られた情報を元に作成した, その楽譜シーケンスにおける音列操作 $O = (o_1, o_2, ..., o_{48})$ の存在の有無 $L = (l_1, l_2, ..., l_{48}) \in \{0, 1\}^{48}$ を用いる. 損失関数には Cross Entropy Loss を用いた. ただし, 学習においては L のほとんどの要素が 0 となるため学習が進みづらい. そこで, 損失関数に L の要素が 0 となる場合と 1 となる場合にそれぞれ 1 : 4 の比率で異なる重みを設定して対策を行う.

条件名	数
曲数	500 曲
声部数	2 声部
声部ごとの音列使用回数	50 回/ 1 曲
和音使用の音数上限	6 音
音列の一部欠如の音数上限	6 音/1 音列
音列の部分入れ替え回数上限	3 回/1 音列

表 1: 楽曲データ生成の条件

条件名	パラメータ数
最大トークン数	256 (512)
埋め込み次元数	256 (768)
Attention-Head 数	4 (12)
Transformer ブロック数	6 (12)

表 2: BERT モデルのパラメータ. 括弧内はオリジナル論文で Devin らの考案した $\mathrm{BERT}_{\mathrm{BASE}}$ におけるパラメータ数

6. 実験

6.1. 実験条件

実験は比較のため, (1) 事前学習・事後学習ともに行った BERT モデル, (2) 事前学習として MLM のみを行い事後学習を行った BERT モデル, (3) 事前学習として NSP のみを行い事後学習を行った BERT モデル, (4)

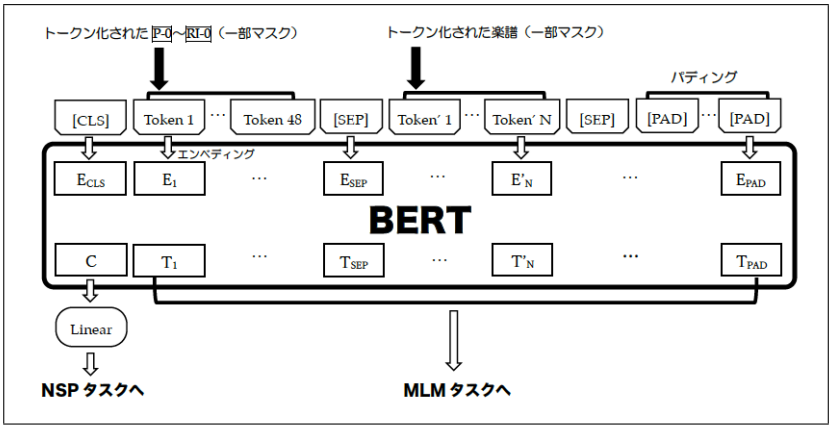

図 3: BERT を用いた事前学習の仕組み

事前学習なしで事後学習のみ行った BERT モデル, および (5) 全結合層のみのモデルを用いた. 事後学習はすべて 100 曲分で行い (5) のみ事前学習に用いた 500 曲分と合わせた 600 曲での学習を行った. ただし (1)〜(3) のモデルでは学習を Transformer ブロック最終層と分類タスクの層のみ行なっており, (4), (5) のモデルではすべてのパラメータについて更新を行なっている. また, それぞれの学習後に, Schönberg の《弦楽四重奏曲第 4 番第 1 楽章》の冒頭 21 小節間でテストを行った.

評価の指標としてはまず, 正解率 Accuracy があげられる. これは各音列操作の予測ごとに正解かどうかを判別し, その平均を取るものであるが, このタスクの場合は, 先述のように正解ラベルのほとんどの要素が 0 となるため, 精度を評価するのに有効な指標となりにくい. そこで, 適合率 Precision, 再現率 Recall を導入する. 適合率 Precision においては, 存在すると予測された音列操作のうち, どの割合が正解であったかを測ることができる. また, 再現率 Recall においては正解データにおいて存在するとされた音列操作をどれだけ予測できたかを測ることができる. ただし, これらはトレードオフの関係にあるため, Precision と Recall の調和平均をとった F1-Score も指標として使用する.

6.2. 実験結果

表 3 に合成データによる実験結果を, 表 4 に実際の楽曲による実験結果を示す. 実験結果から, 事前学習の有無を比較すると, 合成データにおいては (4) の事前学習なしのモデルが一番高い精度であった. これは, (1)〜(3) の BERT モデルと比較して, (4) がすべてのパラメータについて更新を行い, 音列分析のタスクにより特化したモデルを構築できたためだと考えられる. 一方実際の楽曲データによるテストでは, 事前学習を行った (1)〜(3) では Precision, Recall, F1-Score の指標において, 一定数正解を当てることができている. これは事前学習で 500 曲分学習させた 12 音技法の構造などの情報を, 音列分析のタスクに転用できているからだと考えられる.

さらに事前学習の種別で比較すると, 合成データにおいては, (2) の MLM のみの BERT モデルが最も精度が高く, (3) の NSP のみの BERT モデルはすべてのモデルの中で最も低い精度であり, 合成データに限って言えば, 事後学習に NSP が悪影響を与えていると考えられる. 実際自然言語処理の分野においても, NSP タスクは行わなくても精度に影響はない, もしくは行わない方が高い精度が出ることがあるという研究も存在する (Liu et al. 2019). 他方, 実際の楽曲データにおいては MLM, NSP どちらも行った (4) の BERT モデルが最も高い精度を示しており, こうした事前学習の種別とその有効性については, 今後も更なる検証が必要であろう.

7. 結論

本研究では, コンピュータを用いた音列分析の手法として, 合成データを用いた機械学習を提案した. 合成データの生成にあたっては, 音列の部分入れ替えや欠損, さらに複数声部の同時生成等を実装することにより, ある程度の複雑さを持った 12 音音楽を音列ラベル付きで用いることが可能となった. さらに BERT を用いた実験では, 事前学習を行うことにより 12 音技法の内実を学習し, それを事後学習としての音列分析タスクに転用できているらしいことが確認できた.

一方現状では, 実際の楽曲において十分に有用と判断できるモデルを構築するには至っておらず, 楽曲生成手法の更なる多様化・複雑化や, 楽譜情報のトークン化の別の手法の検討, BERT モデルにおけるパラメータの調整, 事前学習種別ごとの有用性の更なる検証などを今後行なっていく予定である. また, 分析対象と同じ音列を使用した合成データからの学習により, 精度が向上する可能性もあるだろう. さらに, 実際の楽曲によ

条件名	Precision	Recall	F1-Score	Accuracy
(1) BERT（事前学習あり）	63.8%	76.3%	66.7%	95.3%
(2) BERT（MLM のみあり）	84.0%	90.9%	85.7%	98.2%
(3) BERT（NSP のみあり）	18.7%	29.0%	21.1%	91.5%
(4) BERT（事前学習なし）	87.4%	96.2%	90.6%	98.6%
(5) 全結合層のみ（参考）	47.9%	69.3%	54.8%	93.3%

表 3: 合成データによる学習結果

条件名	Precision	Recall	F1-Score	Accuracy
(1) BERT（事前学習あり）	15.2%	23.1%	17.0%	93.1%
(2) BERT（MLM のみあり）	8.2%	17.3%	10.4%	90.0%
(3) BERT（NSP のみあり）	1.9%	4.5%	2.6%	94.4%
(4) BERT（事前学習なし）	0.0%	0.0%	0.0%	91.3%
(5) 全結合層のみ（参考）	5.8%	7.1%	6.0%	92.7%

表 4: 実際の楽曲データによるテスト

る学習データを一定量収集できれば, 合成データから実データへのドメイン適応の手法の適用も考えられる.

8. 参考文献

Boulez, P. (1963). *Penser la Musique Aujourd'hui*. Éditions Gonthier.

Chou, Y. et al. (2021). Midibert-piano: Large-scale pre-training for symbolic music understanding. *CoRR abs/2107.05223*.

Devlin, J. et al. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805*.

Leibowitz, R. (1949). *Introduction a la musique de douze sons*. L' Arche.

Liu, Y. et al. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR abs/1907.11692*.

Tanaka, T. and K. Fujii (2014). Melodic pattern segmentation of polyphonic music as a set partitioning problem. In *Proceedings of International Congress on Music and Mathematics*, pp. 291–298.

ZiyuWang and G. Xia (2021). Musebert: Pre-training of music representation for music understanding and controllable generation. In *Proceedings of the 22nd ISMIR Conference*, pp. 722–729.

9. 著者プロフィール

山本 真幸 (YAMAMOTO, Masaki)

北海道教育大学岩見沢校音楽文化専攻作曲コースを経て現在, 東京藝術大学大学院修士課程音楽音響創造研究分野に所属し, P. Boulez など 20 世紀の作曲家における創作技法を研究するほか, AI 技術を用いた音楽創作や分析にも取り組む. これまでに自作品が北海道教育大学岩見沢校定期演奏会や札幌市民芸術祭札幌新人音楽会等に選出, 演奏されたほか, 第 34 回全日本作曲家コンクールソロ部門第 2 位 (1 位なし最高位) 受賞. 作曲を南聡, 阿部俊祐, 田村文生の各氏に師事.

田中 翼 (TANAKA, Tsubasa)

東京藝術大学音楽環境創造科非常勤講師. JST ACT-X 研究員（AI 活用学問革新創生領域）. AI 技芸研究会代表. 専門は音楽情報科学, アルゴリズム作曲, 数理的音楽理論. 京都大学理学部で数学, 東京大学情報理工学系研究科で情報科学を学び, 東京藝術大学先端芸術表現科で博士号を取得. 2014 年より博士研究員として渡仏し, IRCAM, パリ左岸＝ジュシュー数学研究所, ソルボンヌ大学で研究する傍ら音楽院で作曲を学ぶ. 2021 年に帰国し現職.